Featurized Choice Models for Top-K Restaurant Assortment Optimization:

A Comparison of MNL and Machine Learning Approaches

Rishika Gorai   Ryan Gu   Ashlee Liu   Kenny Wongchamcharoen

Fall 2025

1 Introduction & Motivation

Digital food marketplaces like DoorDash face tight interface constraints and can surface only a top-K carousel of restaurants. Choosing this set jointly shapes customer satisfaction (conversion) and platform outcomes (revenue). Standard recommender systems based on traditional machine learning algorithms typically optimize engagement, ranking restaurants by predicted click/buy probability, but this can ignore *cannibalization*, where promoting a cheaper option diverts demand from a higher-value alternative. Revenue management approaches, such as the Multinomial Logit (MNL) model, explicitly model substitution across items, but can be limited by their parametric structure and may miss nonlinear preference patterns. In this project, we study the conversion–revenue trade-off in top-K assortment optimization within a simulated, capacity-constrained, high-intent marketplace, asking whether substitution-aware structural models can outperform the flexible predictive tools that score items independently.

2 Methodology and Experimental Design

To ensure a rigorous comparison, we developed a counterfactual simulation environment governed by a deterministic Ranked Choice Model.

2.1 Groundtruth Dataset Construction

2.1.1 Restaurant Universe

We modeled a fixed universe of $N = 100$ restaurants in Berkeley within a bounded $[0, 10]^2$ coordinate grid. Each restaurant i is characterized by a feature vector containing:

- **Cuisine Type (c_i):** Categorical variable (e.g., Thai, Mexican).
- **Price Level (p_i):** Discrete tier $p_i \in \{1, 2, 3\}$.
- **Rating (r_i):** Continuous variable $r_i \in [3.0, 5.0]$.
- **Location (x_i, y_i):** Coordinates used for distance calculations.

2.1.2 Latent Heterogeneity and Customer Profiles

To capture realistic preference diversity, we generated $P = 10$ distinct customer profiles. Each profile p is defined by a fixed coefficient vector:

$$\boldsymbol{\beta}_p = (\beta_{0p}, \beta_{1p}, \beta_{2p}, \beta_{3p}, \beta_{4p})$$

representing sensitivity to the intercept, cuisine match, price penalty, star rating, and delivery-time (ETA), respectively.

For each simulation instance, a customer u is generated with:

1. A random location (x_u, y_u) sampled uniformly from $[0, 10]^2$.
2. A price tolerance pt_u sampled uniformly from $\{1, 2, 3\}$.

3. A complete ranking of all cuisine categories, generated as a random permutation of the available cuisines (Rank 1 = Most Preferred).
4. An assigned behavioral profile $p(u)$ (e.g., "Budget Shopper," "Speed-Obsessed") sampled uniformly from the 10 available profiles.

2.1.3 Deterministic Utility Model

The groundtruth utility U_{ui}^{true} of restaurant i for customer u is calculated deterministically using the profile weights and specific feature transformations as outlined below:

$$\boxed{\begin{aligned} U_{ui}^{\text{true}} = \beta_{0,p(u)} + \beta_{1,p(u)} \cdot \text{CuisineMatch}_{ui} - \beta_{2,p(u)} \cdot \text{PricePenalty}_{ui} \\ + \beta_{3,p(u)} \cdot \text{RatingNorm}_i - \beta_{4,p(u)} \cdot \text{ETA}_{ui} \end{aligned}}$$

- **CuisineMatch:** A linear normalization of the customer's cuisine rank $r_u(c_i)$:

$$\text{CuisineMatch}_{ui} = \frac{C + 1 - r_u(c_i)}{C}$$

  where C is the total number of cuisines.
- **PricePenalty:** An asymmetric loss function dependent on customer tolerance. Let $g_{ui} = p_i - pt_u$ be the signed price gap. We penalize prices above tolerance more heavily ($\alpha_+ = 1.5 > \alpha_- = 0.5$) than prices below tolerance:

$$\text{PricePenalty}_{ui} = \alpha_+ \max(0, g_{ui}) + \alpha_- \max(0, -g_{ui})$$

- **ETA:** Normalized Euclidean distance relative to the max grid distance ($d_{max} = 10$):

$$\text{ETA}_{ui} = \frac{\sqrt{(x_u - x_i)^2 + (y_u - y_i)^2}}{d_{max}}$$

- **RatingNorm:** Min-max normalization of the 5-star rating:

$$\text{RatingNorm}_i = \frac{r_i - 3}{2}$$

2.1.4 Transaction Data Generation (Training Set)

With the universe and utility model established, we generated the synthetic sales data used to train our models. This process mirrors the observational data a platform like DoorDash would collect:

1. For each customer simulation, we randomly draw an offer set S_u of 5 restaurants from the universe.
2. We compute the groundtruth utility U_{ui}^{true} for all restaurants $i \in S_u$ using the customer's profile-specific β's.
3. The customer then selects the restaurant with the highest utility:

$$t_u = \arg\max_{i \in S_u} U_{ui}^{\text{true}}$$

This yields a dataset of tuples (u, S_u, t_u) which serves as the input for both the MNL estimation and the Machine Learning classifier training.

2.1.5 Key Assumptions

To ensure the validity of our simulation and subsequent analysis, we assume deterministic choice behavior, where customers select the restaurant with the highest utility. Furthermore, we assume a "captive market" where the customer always selects exactly one item from the offered set and there is no "no-purchase" option. We model price sensitivity as heterogeneous, assuming customers penalize prices differently based on their specific tolerance levels. Operationally, we abstract away inventory dynamics, assuming no inventory depletion and no new restaurant entry or exit during the simulation. The platform is constrained to show a fixed assortment size of exactly 5 restaurant options per customer session. We also simplify the menu structure by assuming each restaurant offers a single representative menu item with a single fixed price. Finally, all randomness in the system is assumed to be exogenous.

2.2 Assortment Optimization

2.2.1 Approach 1: Machine Learning Baselines (Random Forest & XGBoost)

To evaluate modern predictive algorithms in assortment optimization, we employed tree-based ensemble methods—specifically Random Forest and Gradient Boosted Trees (XGBoost)—to model customer behavior. Unlike structural economic models, these non-parametric approaches are capable of capturing complex, non-linear interactions between features (e.g., how price sensitivity might interact non-linearly with distance). We trained these classifiers using Python's `scikit-learn` library to predict the individual purchase probability $P(\text{buy} \mid u, i)$ for a given customer-restaurant pair. To address both the class imbalance in the `Buy` vs. `Not Buy` labels and the revenue sensitivity inherent in our objective, we modified the learning procedure for both the Random Forest and XGBoost models. For the Random Forest, we enabled `class_weight='balanced'`, increasing penalties for misclassifying the minority `Buy` class. In addition, for both models, we implemented cost-sensitive learning by weighting each training instance by its base price through the `sample_weight` argument in `fit`. Under this scheme, misclassifying a \$3 item is treated as three times more costly than misclassifying a \$1 item, forcing the models to prioritize accuracy on high-value purchases. We also performed necessary hyperparameter tuning.

Since standard ML classifiers output a probability for each item in isolation, we construct assortments by scoring every restaurant in the universe independently:

- For every restaurant i, we compute the Expected Revenue:

$$\mathbb{E}[\text{Revenue}_{ui}] = \hat{P}(\text{buy} \mid u, i) \times \text{Price}_i$$

- We sort all available restaurants in descending order of their expected revenue.
- We form the final assortment S by selecting the top-K items from this ranked list.

This approach operates under several assumptions that differentiate it from revenue management frameworks. First, it assumes the probability of purchasing from restaurant A depends only on the attributes of restaurant A and the customer, ignoring the other 4 restaurants in the offer set. Second, it treats the assortment problem as a binary classification problem, disregarding the constraint that customers must select exactly one item (substitution effects).

2.2.2 Approach 2: Featurized MNL (Revenue Management)

For our Featurized Multinomial Logit model, we estimated a single global preference vector $\hat{\boldsymbol{\beta}}$ using Maximum Likelihood Estimation (MLE) on the transaction logs. The probability that customer u chooses item i from a given offer set S is modeled as:

$$P_{\hat{\beta}}(i \mid S_u) = \frac{\exp(U_{ui}(\hat{\boldsymbol{\beta}}))}{\sum_{j \in S_u} \exp(U_{uj}(\hat{\boldsymbol{\beta}}))}$$

where $U_{ui}(\hat{\boldsymbol{\beta}}) = \hat{\boldsymbol{\beta}}^\top \mathbf{x}_{ui}$.

To select the top 5 restaurant assortments, the goal is to select a subset $S \subseteq N$ with $|S| \leq K$ to maximize the total expected revenue (with $K = 5$):

$$\max_{S \subseteq N, |S| \leq K} R(S) = \sum_{i \in S} \text{Price}_i \cdot P_{\hat{\beta}}(i \mid S)$$

This objective function presents significant challenges because the revenue function $R(S)$ is generally non-monotone and non-submodular. Adding a cheap, high-utility item (e.g., a burger) can increase total conversion but simultaneously cannibalize sales from high-margin items (e.g., a steak), potentially lowering the total expected revenue of the portfolio. To tackle this, we employ a Greedy Heuristic that builds the assortment by selecting items one by one. Crucially, before adding a new item, it checks if the addition improves the revenue of the entire assortment. This step explicitly resolves cannibalization: if a popular, cheap item 'dilutes' total assortment revenue, the algorithm rejects it. The detailed procedure is described in Algorithm 1. Our MNL model rests on a critical structural assumption that the model estimates a single, global preference vector $\hat{\boldsymbol{\beta}}$ to represent all customers, intentionally averaging out the 10 distinct latent profiles present in the ground truth. This serves to test the model's robustness to misspecification.

Algorithm 1 Greedy Revenue Optimization Heuristic

```
1:  **Input:** customer $u$, Universe $N$, Parameters $\hat{\boldsymbol{\beta}}$, Capacity $K$
2:  **Output:** Assortment $S^*$
3:  Compute $V_{ui} \leftarrow \exp(\hat{\boldsymbol{\beta}}^\top \mathbf{x}_{ui})$ for all $i \in N$
4:  $S \leftarrow \emptyset$, $R \leftarrow 0$
5:  **while** $|S| < K$ **do**
6:      $j^* \leftarrow \arg\max_{j \in N \setminus S} \frac{\sum_{i \in S \cup \{j\}} \text{Price}_i V_{ui}}{1+\sum_{i \in S \cup \{j\}} V_{ui}}$
7:      $R_{\text{new}} \leftarrow \frac{\sum_{i \in S \cup \{j^*\}} \text{Price}_i V_{ui}}{1+\sum_{i \in S \cup \{j^*\}} V_{ui}}$
8:      **if** $R_{\text{new}} \leq R$ **then**
9:          **break**
10:     **else**
11:         $S \leftarrow S \cup \{j^*\}$, $R \leftarrow R_{\text{new}}$
12:     **end if**
13: **end while**
14: **return** $S$
```

2.2.3 Max Utility Oracle Benchmark

We also performed maximum utility optimization as a theoretical ceiling by calculating True Utility ($U_{ui} = \beta_{true} \cdot x_{ui}$) for all restaurants using the customer's groundtruth true deterministic $\boldsymbol{\beta}$, and select the top 5.

3 Evaluation Pipeline and Metrics

3.1 The "Ground Truth Oracle" Framework

We evaluate our optimized assortment by simulating how a real customer would react to them, utilizing the "hidden" knowledge from our Ground Truth generation process. First, our MNL and ML algorithm generates a personalized menu of 5 restaurants (e.g., $S = \{A, B, C, D, E\}$) for each customer. We then pass this menu to the Oracle which looks up the specific customer's true deterministic $\boldsymbol{\beta}$, which was used to generate the ground truth but was hidden from the training phase. The Oracle calculates the exact utility for all 5 items based on the customer's actual $\boldsymbol{\beta}$ (not the model's estimates). It then selects the item with the highest true utility, returning the real revenue collected from that choice. We then compare the three models based on the following performance metrics:

- **Total Revenue:** The sum of the prices of the items actually chosen by the 200 test customers.
- **Average Revenue Per Customer:** Total Revenue divided by the number of customers.

- **Hit Rate:** The percentage of customers who were offered their single absolute favorite restaurant out of the entire universe of 100 options.
- **Average True Utility:** Total Utility divided by the number of customers.

4 Results & Discussion

We evaluated the models on a test set of $N = 200$ customers against the Groundtruth Oracle. Table 1 reports performance metrics for each model.

Metric	**Max Utility Oracle**	**Random Forest**	**XGBoost**	**MNL (Greedy)**
Total Revenue	$358.00	$405.00	$404.00	$524.00
Avg Revenue / customer	$1.79	$2.02	$2.02	$2.62
Global Hit Rate	100%	60.5%	52.0%	16%
Avg True Utility	1.68	1.30	1.30	-0.08

Table 1: Performance metrics evaluated against the Groundtruth Oracle. The "Max Utility Oracle" represents the theoretical ceiling for customer happiness.

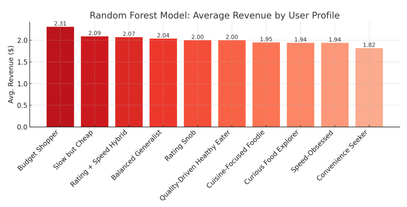
Both Machine Learning baselines achieved high Average customer Utility scores (1.30 for both Random Forest and XGBoost), close to the theoretical maximum of 1.68. This confirms that predictive models act as effective service agents, prioritizing items that closely match customer preferences and maximizing Hit Rate. Indeed, Random Forest achieved a higher Hit Rate (60.5%) than XGBoost (52.0%), though both models captured nearly identical revenue ($404–$405). However, this alignment with customer preference comes at a cost: by independently ranking items and recommending low-margin favorites, ML models systematically underperform on revenue due to cannibalization of higher-value alternatives (Avg. Revenue $2.02). In stark contrast, the MNL model produced a negative Average customer Utility (-0.08), indicating that customers were frequently steered away from their individually optimal choices due to price penalties that exceeded their tolerance. Importantly, this reduction in utility is not a failure of preference estimation but a deliberate consequence of substitution-aware optimization. Under the "No Outside Option" constraint, customers still converted, allowing the MNL model to aggressively upsell toward higher-margin substitutes. This strategy extracted consumer surplus rather than improving preference matching, driving Average Revenue to $2.62 per customer, a 46% increase over the Max Utility baseline, despite substantially lower Hit Rates. However, this revenue figure represents an upper bound; in non-captive markets, such negative utility would likely trigger churn rather than forced conversion.

4.1 Profile-Level Analysis

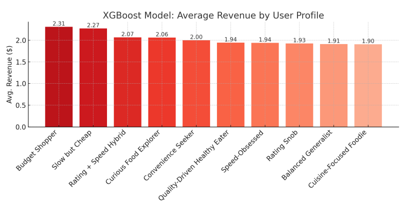
To understand how each model handles customer heterogeneity, we disaggregated average revenue by customer profile. The MNL model exhibits high variance, extracting maximum revenue ($\approx$ $3.00) from price-insensitive profiles like "Cuisine-Focused Foodie" and "Curious Food Explorer," while extracting less from "Budget Shoppers." This indicates successful segmentation and targeting. In contrast, the ML models perform relatively well for price-sensitive customers like "Budget Shopper" and "Slow but Cheap" but show a relatively worse revenue distribution ($\approx$ $2.00) across all the other profiles, failing to capture the potential surplus from high-willingness-to-pay customers as shown in Figure 1.

4.2 Revenue Gap Analysis

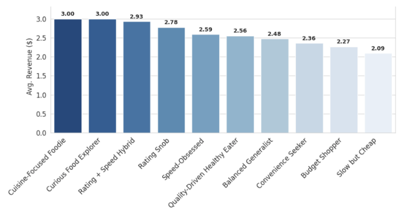
We define the "Revenue Gap" as the difference between the actual revenue captured and the revenue of the customer's true favorite item (Gap = Actual – Favorite). Figure 2 shows the distribution of Revenue Gap between the two models.



(a) Random Forest



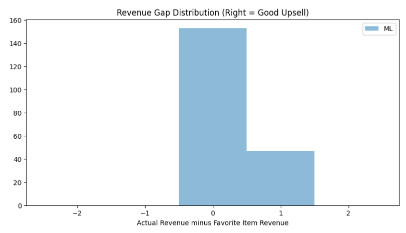
(b) XGBoost



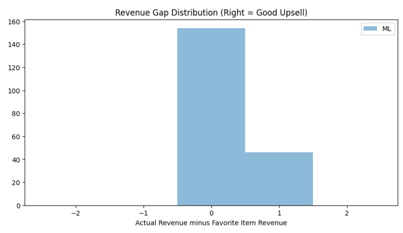
(c) MNL

Figure 1: Average Revenue per customer by customer profile. MNL aggressively targets high-value customers, while ML provides consistent but lower-value recommendations.

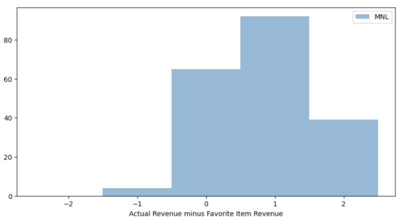
- **Gap ≈ 0:** The model sold the customer exactly what they wanted. This was predominantly observed in the ML results, which prioritized Hit Rate.
- **Gap > 0 (Upsell):** The model successfully forced the customer to buy a more expensive substitute. This positive gap was a key feature of the MNL results, confirming that the low Hit Rate is a deliberate feature of successful yield management, not a bug.



(a) Random Forest



(b) XGBoost



(c) Featurized MNL

Figure 2: Distribution of Revenue Gap for ML vs. MNL Models. ML distributions (a,b) are centered near zero, while MNL (c) is shifted right, reflecting upselling behavior.

5 Implications and Conclusion

Our findings highlight a strategic pivot for digital marketplaces under peak demand. In high-intent scenarios (e.g., Friday 8 PM), customer demand is effectively captive and price-inelastic, shifting the platform's objective from conversion to monetization. Standard machine learning recommenders, optimized for engagement and "easy wins," inadvertently dilute revenue by cannibalizing high-margin options with low-value favorites. In contrast, MNL-based assortment optimization acts as a guardrail against this dilution. By selectively withholding low-margin items, the model induces substitution toward higher-margin alternatives, reducing Hit Rate but maximizing revenue from captive demand. More broadly, optimization of top-K assortment should focus not on what customers *prefer*, but on what they are willing to *accept* at the highest value. Ultimately, superiority of each model depends on the metric. While ML models are better suited for low-demand settings where retention matters, revenue management models that take into account substitutions provide a clear advantage in high-demand environments. Future work could explore hybrid approaches, such as latent-class MNL or multi-objective optimization, to balance long-term conversion with short-term revenue extraction.